How to Make Web Sites Talk Together – Web Service Solution

Hoang PHAM HUY
Hanoi University of Technology
Tel.: 84-4-8680896
hoangph@it-hut.edu.vn

Takahiro KAWAMURA
Toshiba R&D Center
Tel.: 81-44-549-2237
takahiro@isl.rdc.toshiba.co.jp

Tetsuo HASEGAWA
Toshiba R&D Center
Tel.: 81-44-549-2237
tetsuo3.hasegawa@toshiba.co.jp

ABSTRACT

Integrating web sites to provide more efficient services is a very promising way in the Internet. For example searching house for rent based on train system or preparing a holiday with several constrains such as hotel, air ticket, etc... From resource view point, current web sites in the Internet already provide quite enough information. However, the challenge is these web sites just provide information but do not support any mechanism to exchange them. As a consequence, it is very often that a human user has to take the role to "link" several web sites by browsing each one and get the concrete information. The reason comes from a historical objective. Web sites were developed for human users browsing and so, they do not support any machine-understandable mechanism.

Current researches in WWW environment already propose several solutions to make newly web sites become understandable to other web sites so that they can be integrated. However, the question is how to integrate existing web sites to these new one. Evidently, redeveloping all of them is an unacceptable solution. In this paper, we propose a solution of Web Service Gateway to "wrap" existing web sites in Web services. Thus, without any efforts to duplicate the Web sites code, these services inherit all features from the sites while can be enriched with other Web service features like UDDI publishing, semantic describing, etc…

This proposal was developed in Toshiba with Web Service Gateway and Wrapper Generator System. By using these systems, several integrated-applications were built and they are also presented and evaluated in this paper.

Categories and Subject Descriptors

D.2.13 [Reusable Software]: Domain engineering, Reusable libraries, Reuse models.

General Terms: Design, Standardization, Theory.

Keywords

Web Service, WSDL, Service Development, Wrapper, Web site

1. Introduction

The success of the Internet does not only allow the connection of computers and business partners world-wide but also open a new way to carry out the business transactions. Supply and commerce over the Internet, such as online weather forecast ([9]) or online book shops ([10][11]), have already entered to the market as individual information sources. Broadly saying, "Business-to-Customer oriented" and "not-open", are the two principle characteristics of almost all theses information sources. The latter can be considered as a consequence of the former since supporting Business-to-Customer schema does not require one information source to be opened to others. However, Business-to-Business must be a considerable market in the future e-business in which each provider needs to share its capabilities with others. This future market also requires an infrastructure to integrate several providers in a global service, in supporting other service-independent features like accounting, billing, security, etc…. Evidently, Web Service technology is the most promising candidate. The question that we want to tackle in this paper is related to the existing information sources in the current Internet. That is how to harmonize them with the future Business-to-Business market ?

A common way to "import" the existing information sources to a new market is installation of wrapper components that act as representatives of the "old" providers. Though that it is not a new research domain, adopting this wrapper mechanism in our approach under the Web Service strategy turns it out as a promising solution. Thanks to the current efforts of developing and standardizing Web Service technology, our Web Service gateway brings all the advanced features of Web Service to the current existing Internet information source, without paying much efforts to re-developing them. Keeping in mind that more than 80% current information sources in the Internet realized on dynamic Web sites with a underlying database ([1]), we concentrate to support this kind of Web sites. Figure 1 shows general components of our Web Service gateway and the way to import existing Internet information sources to Web Service domain. Because Internet information sources differ from each other in the way to access (protocol, query structure, etc…) and in the format of returned HTML pages, a different wrapper is required to represent each information source. Theses wrappers are generated with the help of Wrapper Generator System and automatically deployed to the gateway. They take the responsibility to

- receive requests from service clients in Web Service domain;
- convert the request to an appropriate form before sending to the existing information source;
- get the returned HTML pages and extract the required concrete data;
- return the data to service client in Web Service.

WWW 2005, May 10-14, 2005, Chiba, Japan.
ACM 1-59593-051-5/05/0005.

Theses tasks are carried out by using several appropriated supports from the Generic Wrapper System, available in the gateway. In order to support different technologies in existing Internet sources as well as in Web Service domain, plugin was adopted as designing strategy in our Web Service gateway. For instance, if we need to support a new kind of information source, a new plugin can be added in the Generic Wrapper System and providing necessary library for newly wrappers.

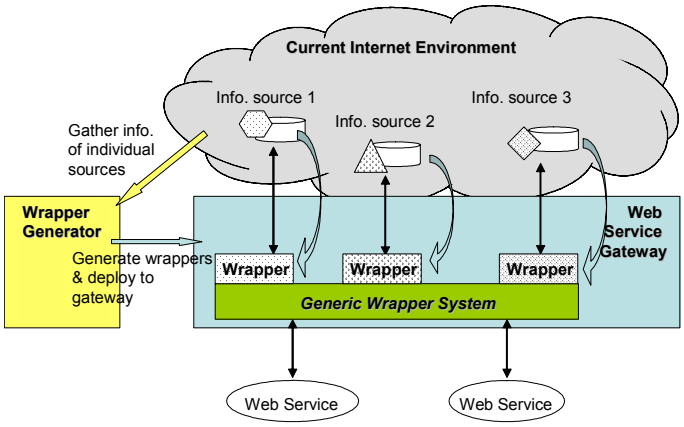

Figure 1: Toshiba Web Service Gateway

In the next section, we briefly describe several related works in pointing out several related aspects in our approach. Then, section 3 and 4 describe in detail the architecture and implementation of our Web Service gateway. Section 5 presents a demonstration of using out Web Service gateway to create an e-business service from several existing Internet information sources. The last section concludes our work and draw our several future works.

2. Related Work and Our Approach

The wrapper idea has been considered for several years. Basic mechanism is the same but in each period of time, the applied method were different according to the current trend of technology. In Stanford's TSIMMIS project on 1997 (**[4]**), wrapper was used to provide an universal access mechanism for heterogeneous information source including database, Web site, etc… The key point of TSIMMIS is not creating wrapper it-self but customizing wrapper's interface, based on a concept of wrapper-template. For that, TSIMMIS provides several "hard-code" wrappers for different information sources. Theses wrappers, when being used in a particular application, can be customized with newly defined wrapper-templates in order to provide an appropriated interface for the application. This approach has a weakness is that each information source requires a new "hard-code" wrapper and moreover, wrapper generation was completely lacked in TSIMMIS. However, the idea of wrapper-template is a strong point and it is inherited in our approach to automatically generate wrapper, based on some definitions of users.

Jedi (**[6]**) is another project organized in German some years after TSIMMIS. Within objective to mostly support information sources of Web site, Jedi concentrates to the procedure of parsing and extracting data in a HTML page. For that, Extraction Language was defined to pattern data in a HTML page and then, a Jedi parser (provided as several Java libraries) can be used to extract the data in the patterned HTML page. Like TSIMMIS, Jedi also did not consider the aspect of wrapper generation. It just provide a mechanism and several Java libraries to create HTML parser and extract concrete data in a HTML page. By taking the design approach of plugin, out Web Service gateway can apply Jedi mechanism to create a Intelligent HTML Parser. It will be presented in section 3.2.

Automatically wrapper generation was considered in UMICAS (**[3]**). Qualified-path-expression Extractor Language (QEL) and Complex Extractor Specification Language (CESL) were used to define the query that wrappers should use to extract data in a HTML page. User can examine a HTML page and use a GUI toolkit to define query with theses two language. Then, a wrapper can be automatically generated by just one mouse click in the GUI toolkit. Although this project had quite completely covered the Web site wrapper domain, standardization aspect is its weak point. QEL and CESL were not widely accepted as a standard method for extracting data in HTML page. The generated wrappers also did not provide a standard interface to receive client's request. Nowadays, XPath (**[12]**), XQuery (**[13]**) and DOM (**[14]**) specifications are widely accepted in place of QEL and CESL. For wrapper's interface to receive request, Web Service Description Language (WSDL) is world-wide accepted specification. That's why they are taken into account in our approach.

Semi-automatic wrapper generation ([2], [7], [8]) can be considered as the most advanced wrapper generation mechanism currently. User with the help of a GUI can analyze a HTML page and define several data extraction rules. The rest of wrapper generation procedure is delegated to a to toolkit. This "semi-automatic" way can support complicated tasks in seeking data of a HTML page by the help from users while releasing them from several common uninteresting routines. However, theses works all try to define their own descriptive language in order to define the data extraction rules which is, we think, not a intelligent approach. This strategy implicitly excludes the possibility of applying other mechanisms to extract data from HTML page. Keeping in minds this observation, our approach also follows semi-automatic wrapper generation mechanism but try to define a common framework for Intelligent HTML Parsers, which are the plugins in our Web Service gateway. This framework provide a common interfaces that wrappers can use to extract data in HTML pages. On the other hand, each plugin can be implemented with different data extraction mechanism, based on different descriptive languages to pattern data in HTML pages. Users can use our plugin discovery tool to find out which plugin parser is the most appropriate for the HTML page pattern of a particular Web information source, then "bind" this plugin to the corresponding wrapper. The wrapper description it-self is not changed. This is the "intelligence" aspect in our HTML parser framework. Concerning the characteristic of "most appropriate" parser, wrapper verification concept was also proposed in [5].

3. Web Service Gateway and Web Service Wrapper

As generally presented in figure 1, our Web Service gateway consists of two principle parts – the Web Service Wrappers and the Generic Wrapper System. Theses components will be discussed in the following sub-sections.

3.1 Web Service Wrapper

The logical design of our Web Service wrappers can be seen in figure 2. Each wrapper consists of 4 basic modules:

- *Web Service Interface*. This interface is defined with WSDL, providing an entry point to access to the wrapper. Through this interface, other entities in Web Service domain see and make use of this wrapper as a normal Web Service. Evidently, other Business-to-Business supported mechanisms in future Web Service domain such as UDDI, ontology-based service seeking, etc… can be applied with these Web Service wrappers.
- *Wrapper Coordinator*. This module takes the responsibility to transfer the requests received through Web Service interface into another form that the Internet information source can understand. For example, for wrapping CGI Web sites, the Wrapper Coordinator module coverts the parameters' value in Web Service interface into a CGI query conforming to the CGI Web site.
- *Information Source Access Protocol*. This module provide an engineering feature to transport the requests, after being harmonized by Coordinator, to the information source. It also takes the responsibility to retrieve the HTML pages, which is the reply from information source. Depended on the technique that the Internet information source uses, the corresponding accessing protocol is applied in this module by a driving from Coordinator. For example CGI GET, CGI POST or ASP, JSP, etc…
- *Data Extractor*. This module takes the responsibility to extract data from HTML pages and construct an appropriate data structure. This data structure is then returned to the requesting entity in Web Service domain, through Web Service Interface. Instead of hard-code integrating HTML data extraction mechanism, this module communicate with several independent HTML parsers (the Parser plugin in Web Service gateway) to carry out its task.

The key point in our approach is that the wrappers are not hard-coded in Web Service gateway but they are automatically generated by Wrapper Generator tool, based on a wrapper description. This description contains all necessary information of the Internet information source (accessing protocol, request format, etc…), the signature of the Web Service interface as well as the mapping between theses two items. By designing wrappers with several independent basic modules and supporting modules generation feature, our Web Service gateway can be easily adapted to new business environment or different kinds of information source. For example, to provide a gateway interface for J2EE Enterprise Java Bean environment, the wrapper description files can be changed and a newly wrapper's class can be generated with EJB interface in the place of Web Service Interface

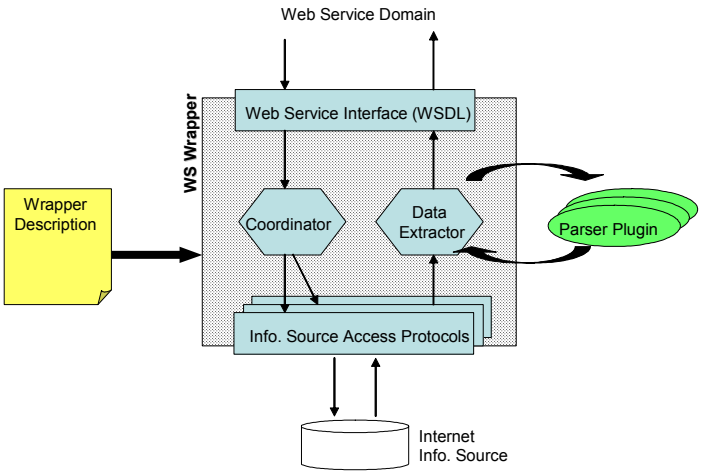

Figure 2: Web Service Wrapper

3.2 Intelligent HTML Parser Plugin

As described above, data extraction from HTML pages is carried out by several intelligent HTML parsers. Theses parsers are deployed in the Generic Wrapper System of the Web Service gateway as the plugin entities. By adopting this design strategy, any third parties can develop their own HTML parsers, based their own mechanism, and deploy to our Web Service gateway to make them available to the wrappers. For that, we need to define a standard framework that all HTML parsers implementation must follow to be able to execute in our gateway. The following two mandatory requirements are strongly considered in our framework design:

- The algorithm to extract data from a HTML page and the way to implement it should be as flexible as possible so that third parties can freely (as much as possible) decide the way to develop their intelligent HTML parser.
- One standard communication mechanism must be designed so that all HTML parsers must support to communicate with the Web Service wrappers.

For the first requirement, as the fact that all HTML pages are constructed by a tree-style HTML tags, we propose to abstract all HTML pages format to some kinds of tree-styles nodes. Thus, the path to locate data in a HTML page can be abstract to:

root.node(1).node(2)…node(n)

We call this is a path-finder sentence which consists of several node-finder elements. Each node-finder element (node(1), node(2), etc…) can be constructed by an explicit node address such as root node or a relative address in comparing with the node parent. The "intelligence" in this approach is that even the abstract HTML page format if fixed to tree-style but the realization of theses trees are completely depended on the implementation. So does the node-finder. Depended on the parser implementation mechanism, a node-finder can be an explicit digit position like "1.0.3.5" or a textual position expression like "second DIV tag" or "first DIV tag after '1.0.3.5'". Figure 3 shows that the same HTML page can be represented in two different tree-style structure with two different path-finder sentence to locate the same data.

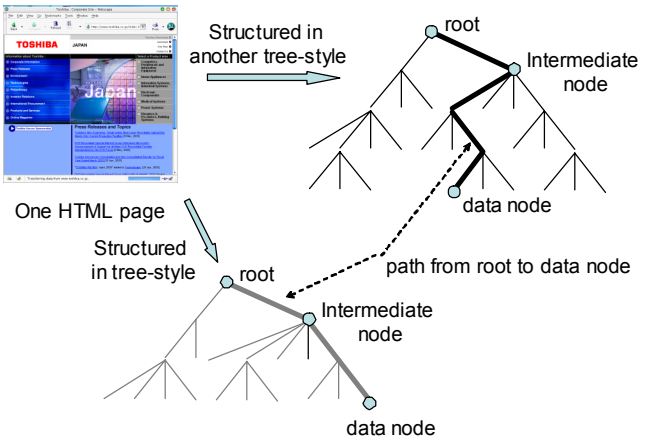

Figure 3: Parsing a HTML page by flexible tree-style

Thus, the procedure to extract data from HTML page become is to find out the path-finder sentence. In more detail, it is a sequence of information exchange between the wrapper's Data Extractor module and an appropriate HTML parser plugin to, firstly, construct a tree-style structure and, secondly, setup the next node-finder, and so on. Evidently, the information that the wrapper's Data Extractor module sends to the HTML parser to establish the path-finder sentence must be conformed to the parser implementation. This information is given to the wrappers when they were generated and deployed to the Web Service gateway, by its wrapper description.

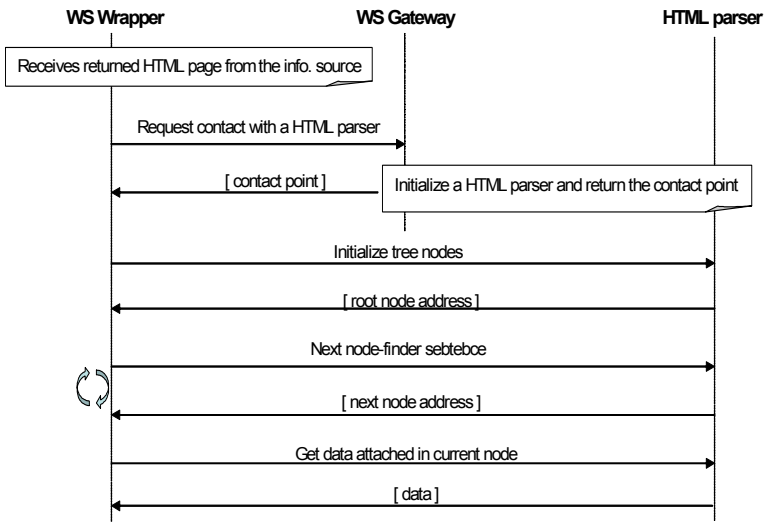

Figure 4: Standard Communication Framework

Figure 4 presents the standard communication framework between wrappers' Data Extractor module and HTML parser plugin to extract data. After receiving a HTML page from the information source, the wrapper's Data Extractor module contacts with the Web Service gateway to initialize an appropriate HTML parser. After initializing the HTML parser, Web Service gateway returns the parser reference to the Web Service wrapper. By invoking several standardized operation to this parser reference, the wrapper requests the parser to initialize a tree-style nodes represented to the current HTML mode. Then, the wrapper invokes one of several time to the parser to send the node-finder sentences as defined in the wrapper description to find the data node. As explained before, it is depended on the parser implementation that the node-finder can be various, such as "go to the first image node" or "go to the second children node". Finally, the wrapper requires the HTML parser send back the data in current data node.

Based on above designing strategy, we define a set of operations that any intelligent HTML parsers must provide to be invoked by the wrappers. The most important plugin function is getNodePosition():

getNodePosition(parrentPosition, node-finder)

This function calculates and returns the position of the current node in the tree, based on the parent node and the relative "distance" to the given node. The position format of these node (parrent, current and node-finder) are completely depended on the plugin implementation. For example, one plugin accepts an explicit node tree-based digit position like "1.0.3.5" while another can provide a plugin which accepts both above explicit digit position and a textual position expression like "second DIV tag" or "first DIV tag after '1.0.3.5'". XPath, XPointer or XQuerry can be applied here to define the node-finder sentence. Thus, the more human-like in position format that a plugin HTML parser accepts, the more intelligent in the parser to extract data.

4. Wrapper Generator Toolkit

In order to support users to define the wrapper description and to generate wrapper code, Web Service wrapper generator toolkit was developed in Toshiba R&D Center. This toolkit consists of 2 main components - Wrapper Generator and Plugin Discovery.

Figure 5 presents the main windows of our Web Service Wrapper Generator (WSGen). This tool supports a GUI to analyze information source to define wrapper description. Then, by just one mouse click, the wrapper code is generated and deployed to Web Service gateway. In the first version, we concentrate to the Internet information source based on CGI technology. For that, by enter a normal CGI query, including the protocol, CGI Web site address, resource path and the query pattern, WSGen automatically generate a primitive Web service for the given CGI Web site. Web Service interface and CGI mapping is carried out by several steps, in several working tab. In each tab, user can graphically create, modify, delete, etc… the elements of the wrapper description. After generating a Web Service wrapper, user can go to "Deploy working tab" in WSGen to deploy it to the Web Service gateway. Toshiba provides a default Web Service gateway that is delivered with WSGen. Otherwise, user can take the generated Web Service gateway (packed in a WAR file – Web Application Archives) to deploy it in any other Web Service hosting systems. This WAR file is packed with all necessary library such as accessing protocol, HTML parser plugin, etc… so it can be deployed to any Web Service hosting server which support WAR specification.

Figure 6 presents the main windows of our Plugin Discovery. This tools allows user to try all the HTML parsers available in the Web Service gateway to find which is the most suitable for parsing HTML pages returned from a given information source. This tool

also allows user to define the node-finder sentences. After selecting an available HTML parser, the tree nodes is shown to the user. When user click to a node in HTML page, the discovery engine will analyze the data attached to this node and propose all the possible code-finder sentence to go to this node.

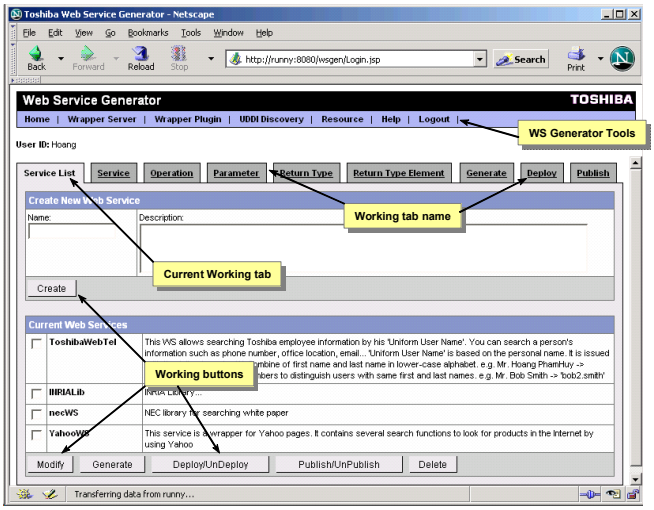

Figure 5: Web Service Wrapper Generator main windows

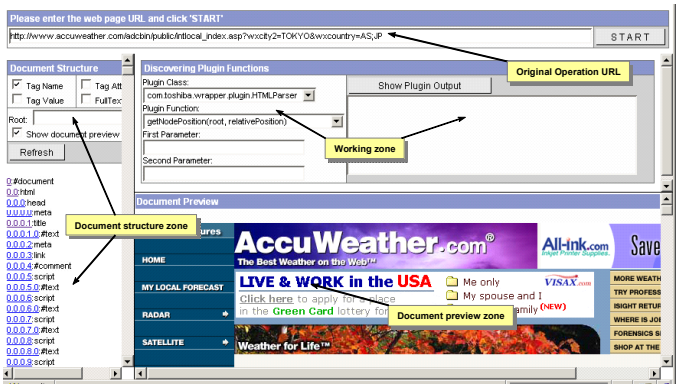

Figure 6: Plugin Discovery Tool

5. Applying Web Service Gateway

By using Toshiba WSGen, several wrappers were generated for certain CGI Web sites like Yahoo, Amazon, etc… and deployed to Toshiba Web Service gateway. This gateway provides Web Service interfaces inside our group in order to test the feasibility of several Web Service in future business environment. In this section, we introduce one of our applications based on this system.

Currently, there are a lot of Web sites selling books in the Internet (book-online Web site). By entering a book's ISBN number, user can see the price in a Web page. The idea is creating a service to buy a book with the best price. Figure 7 presents the general architecture of this application.

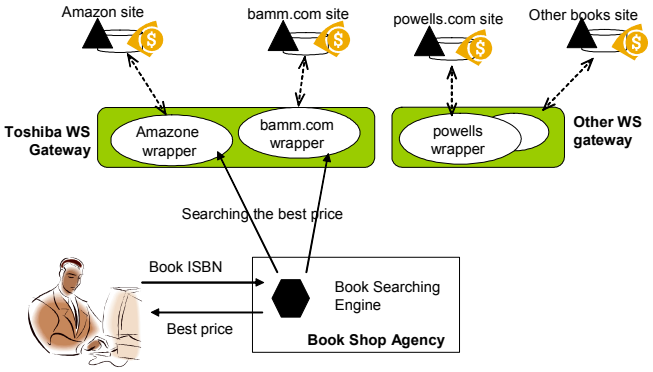

Figure 7: Book shop agency application

Amazon, Bamm, Powells, Textbook and Barnes & Noble are the book-online CGI Web site that were taken in to account in this example. For example, to view the price of "Reading in AGENTS" – ISBN 1558604952, the following CGI queries can be sent to the corresponding sites:

- http://www.amazon.com/exec/obidos/ASIN/1558604952
- http://www.booksamillion.com/ncom/books?type=isbn_search&find=0201485435
- http://www.powells.com/cgi-bin/biblio?isbn=1558604952
- http://www.textbookx.com/product_detail.php?detail_isbn=1558604952
- http://search.barnesandnoble.com/textbooks/booksearch/isbninquiry.asp?isbn=1558604952

By using WSGen toolkit, five Web Service wrappers for these five online-book sites are generated. They are deployed in several Web Service gateways and published to an UDDI registry. Each Web Service wrapper provides an operation getPrice(isbn) with one parameter – the book's ISBN number. The returned value of these wrapper operations is the price of the corresponding books which is displayed in the corresponding Web site. Making use of these wrappers, a Book shop service was developed. In order to search the best price, user gives the ISBN to a book search agent. After sending several message to these book-online wrappers, agent receive a list of price. It then calculates and gives the best one to the user.

6. Conclusion and Future Works

The Book Shop Agency application is not the first "best price book search" in the Internet. There are, for example isbn.nu (**[11]**), several web sites providing such kind of price comparing service. However, underlying mechanism of these sites is propriety. It can have even some kinds of private interaction with the other web sites to directly access to their database. By applying our Web Service gateway, we provide a standard mechanism for Business-to-Business with existing Internet resources. This product of Web Service gateway have been completed in our laboratory and started to be distributed in the market.

Currently, the Toshiba Web Service gateway is being upgraded within several directions. The first one is to integrate Toshiba WSGen with MatchMaker, a Web service match making system supporting semantic description. It will allow Web Service

wrappers, after being generated, to be automatically published with semantic description. The second direction is supporting other interfaces than Web Service, such as EJB. Another future work is to develop new intelligent HTML parser allowing wrapper to perform more human-like actions.

References

[1] A. Sahuguet and F. Azavant, "Building Light-Weight Wrappers for Legacy Web Data-Sources Using W4F", *Proceedings of 25th International Conference on VLDB*, 1999, pp. 738-741

[2] R. Baumgartner, S. Flesca, and G. Gottlob, "Visual Web Information Extraction with Lixto", 2001, *Proceeding of 27th Conference on VLDB*, pp. 119-128

[3] J. Gruser, L. Raschid, M. Vidal, and L. Bright, "Wrapper Generation for Web Accessible Data Sources", *Proceeding of Conference on Cooperative Information Systems*, 1998, pp. 14-23

[4] J. Hammer, H. Garcia-Molina, S. Nestorov, R. Yerneni, M. Breunig, and V. Vassalos, "Template-Based Wrappers in the TSIMMIS System", *Proceedings of 23rd ACM SIGMOD Conference on Management of Data*, 1997

[5] K. Nicholas, "Wrapper verification", *World Wide Web*, Kluwer Academic Publishers, Volume 3, Issue 2, 2000, pp. 79-94

[6] G. Huck, P. Frankenhauser, K. Aberer, and E. Neuhold, "Jedi: Extracting and Synthesizing Information from Web", *Proceeding of 3rd Conference on Cooperative Information Systems*, 1998, pp. 32-43.

[7] L. Lui, C. Pu, and W. Han, "An XML-Enabled Wrapper Construction System for Web Information Sources", 2000, *Proceeding of 15th Conference on Data Engineering (ICDE)*, pp. 611-621.

[8] M. Christoffel, B. Schmitt, and J. Schneider, "Semi-Automatic Wrapper Generation and Adaption", *Proceeding of Conference on Enterprise Information Systems*, 2002

[9] AccuWeather: *http://www.accuweather.com*

[10] Amazon book shop: *http://www.amazon.com*

[11] Isbn.nu book shop: *http://www.isbn.nu*

[12] XPath Specification: *http://www.w3c.com/TR/xpath*

[13] XQuery Specification: *http://www.w3c.org/XML/Query*

[14] W3C Document Object Model (DOM): *http://www.w3c.com/DOM*